加微信：642945106 发送“ 赠送 ”领取赠送精品课程
发数字“ 2 ”获取众筹列表

11 | Gossip协议：流言蜚语，原来也可以实现一致性

2020-03-06 韩健

讲述：于航
时长 10:07 大小 9.28M

你好，我是韩健。

有一部分同学的业务在可用性上比较敏感，比如监控主机和业务运行的告警系统。这个时候，相信你希望自己的系统能在极端情况下（比如集群中只有一个节点在运行）也能运行。回忆了二阶段提交协议和 Raft 算法之后，你发现它们都需要全部节点或者大多数节点正常运行，才能稳定运行，那么它们就不适合了。而根据 Base 理论，你需要实现最终一致性，怎么样才能实现最终一致性呢？

在我看来，你可以通过 Gossip 协议实现这个目标。

Gossip 协议，顾名思义，就像流言蜚语一样，利用一种随机、带有传染性的方式，将信息传播到整个网络中，并在一定时间内，使得系统内的所有节点数据一致。对你来说，掌握这个协议不仅能很好地理解这种最常用的，实现最终一致性的算法，也能在后续工作中得心应手地实现数据的最终一致性。

为了帮你彻底吃透 Gossip 协议，掌握实现最终一致性的实战能力，我会先带你了解 Gossip 三板斧，因为这是 Gossip 协议的核心内容，也是实现最终一致性的常用三种方法。然后以实际系统为例，带你了解在实际系统中是如何实现反熵的。接下来，就让我们开始今天的内容吧。

Gossip 的三板斧

Gossip 的三板斧分别是：直接邮寄（Direct Mail）、反熵（Anti-entropy）和谣言传播（Rumor mongering）。

直接邮寄：就是直接发送更新数据，当数据发送失败时，将数据缓存下来，然后重传。从图中你可以看到，节点 A 直接将更新数据发送给了节点 B、D。

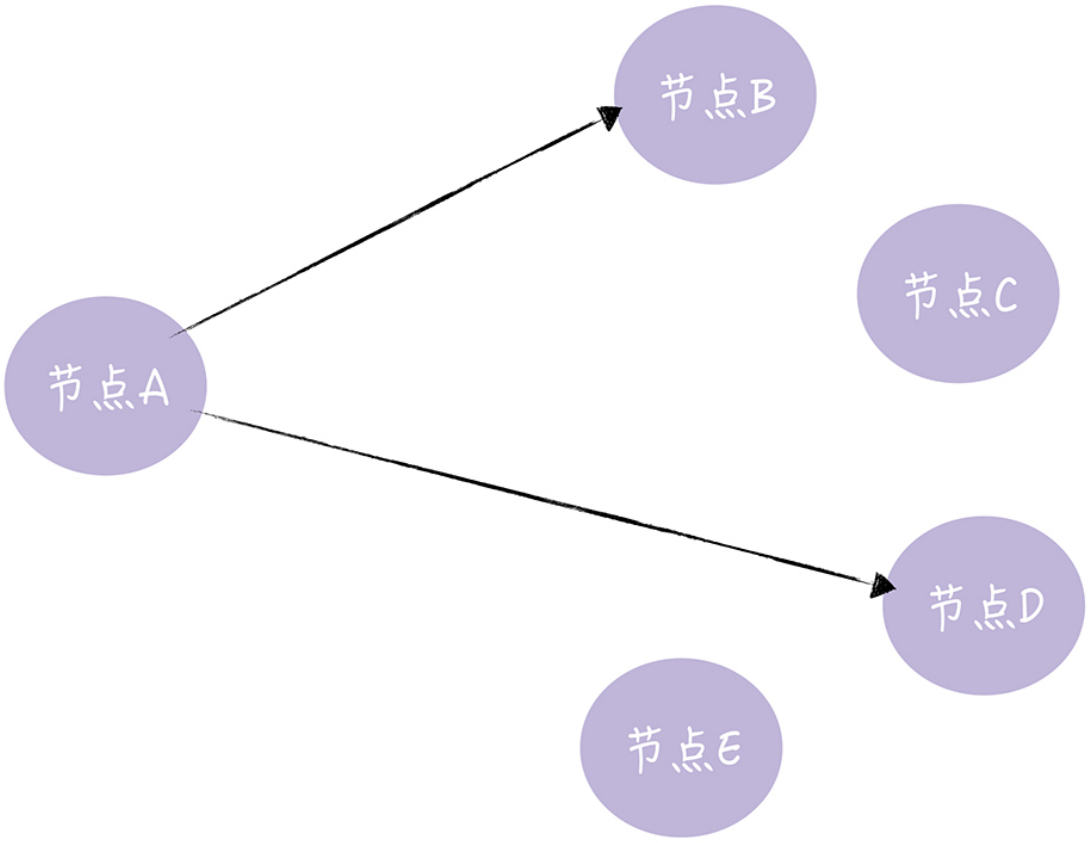


图1

在这里我想补充一点，直接邮寄虽然实现起来比较容易，数据同步也很及时，但可能会因为缓存队列满了而丢数据。也就是说，只采用直接邮寄是无法实现最终一致性的，这一点我希望你能注意到。

那如何实现最终一致性呢？答案就是反熵。本质上，反熵是一种通过异步修复实现最终一致性的方法（关于异步修复，你可以回顾一下04 讲）。常见的最终一致性系统（比如 Cassandra），都实现了反熵功能。

反熵指的是集群中的节点，每隔段时间就随机选择某个其他节点，然后通过互相交换自己的所有数据来消除两者之间的差异，实现数据的最终一致性：

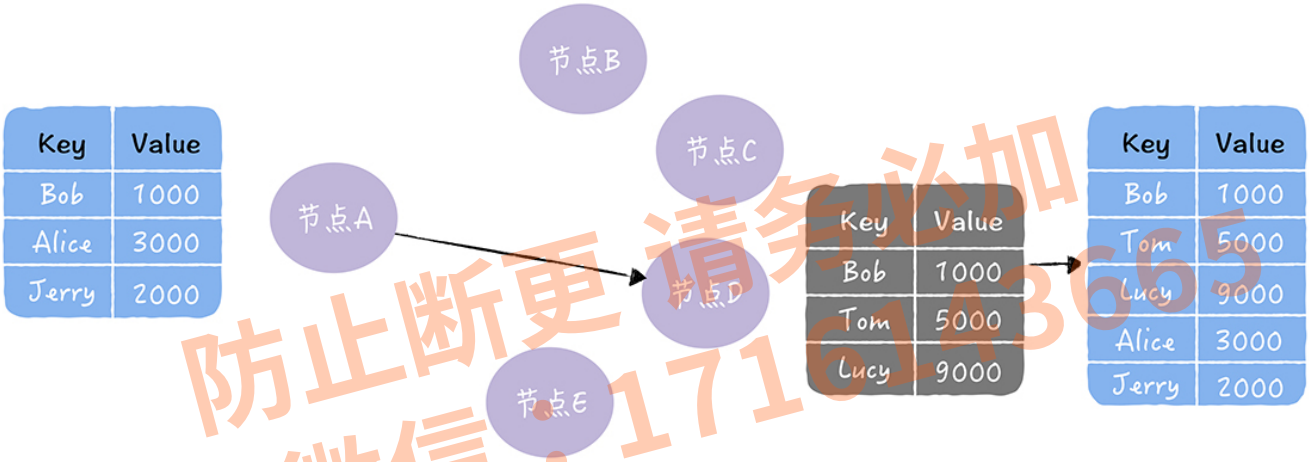


图2

防止断更 请务必加
首发微信：17164243665

从图 2 中你可以看到，节点 A 通过反熵的方式，修复了节点 D 中缺失的数据。那具体怎么实现的呢？

其实，在实现反熵的时候，主要有推、拉和推拉三种方式。我将以修复下图中，2 个数据副本的不一致为例，具体带你了解一下。

图3

推方式，就是将自己的所有副本数据，推给对方，修复对方副本中的熵：

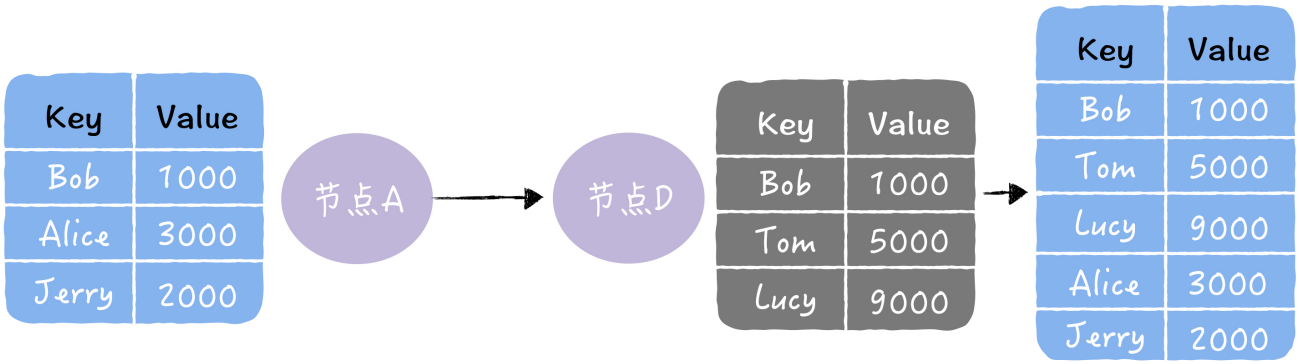


图4

拉方式，就是拉取对方的所有副本数据，修复自己副本中的熵：

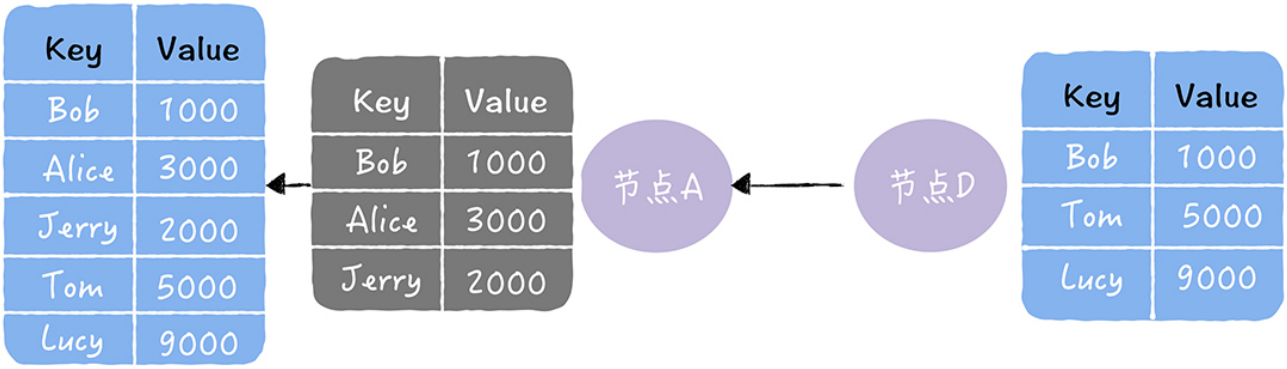


图5

理解了推和拉之后，推拉这个方式就很好理解了，这个方式就是同时修复自己副本和对方副本中的熵：

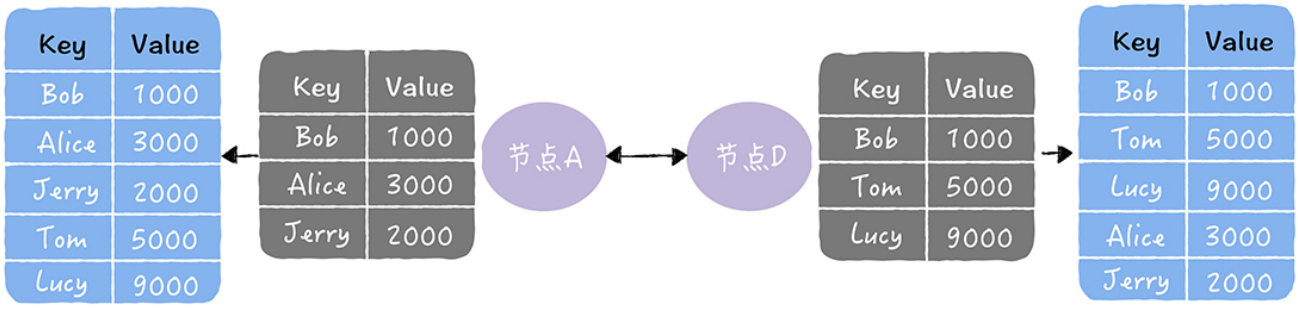


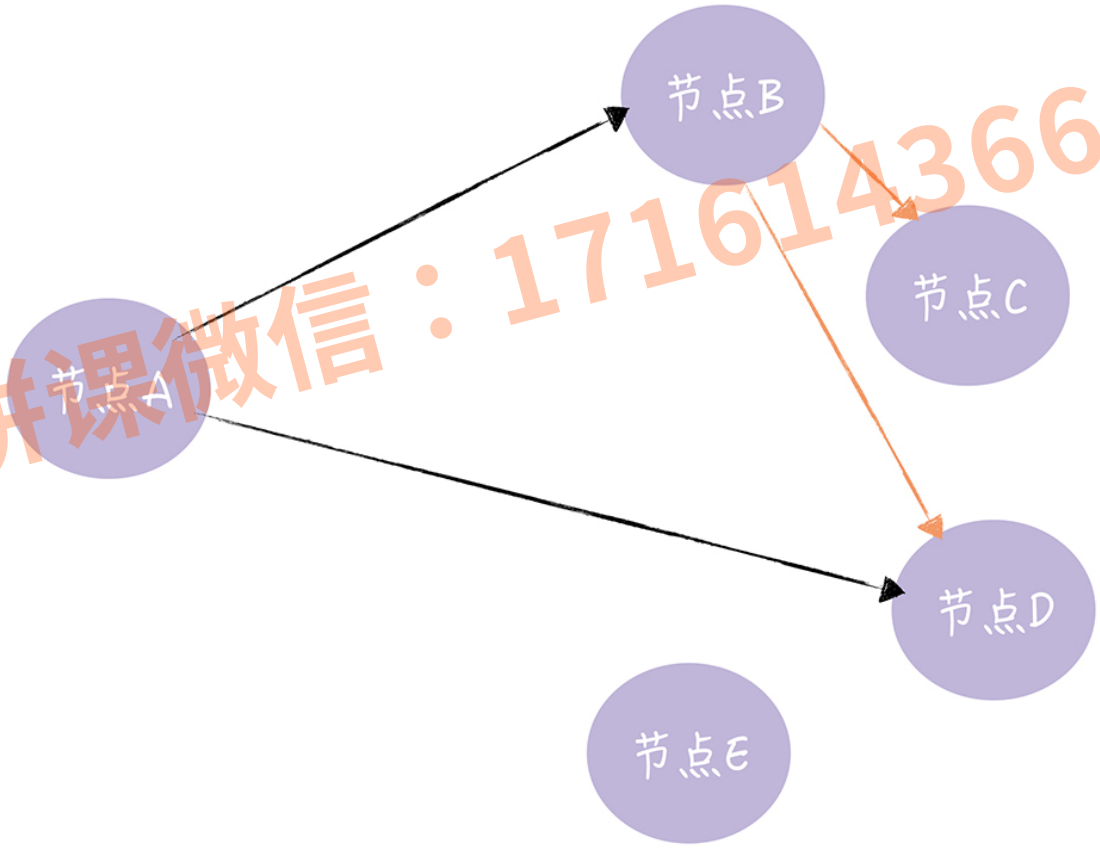
图6

也许有很多同学，会觉得反熵是一个很奇怪的名词。其实，你可以这么来理解，反熵中的熵是指混乱程度，反熵就是指消除不同节点中数据的差异，提升节点间数据的相似度，降低熵值。

另外需要你注意的是，因为反熵需要节点两两交换和比对自己所有的数据，执行反熵时通讯成本会很高，所以我不建议你在实际场景中频繁执行反熵，并且可以通过引入校验和（Checksum）等机制，降低需要对比的数据量和通讯消息等。

虽然反熵很实用，但是执行反熵时，相关的节点都是已知的，而且节点数量不能太多，如果是一个动态变化或节点数比较多的分布式环境（比如在 DevOps 环境中检测节点故障，并动态维护集群节点状态），这时反熵就不适用了。**那么当你面临这个情况要怎样实现最终一致性呢？答案就是谣言传播。**

谣言传播，广泛地散播谣言，它指的是当一个节点有了新数据后，这个节点变成活跃状态，并周期性地联系其他节点向其发送新数据，直到所有的节点都存储了该新数据：



拼课微信：17161436665

图7

从图中你可以看到，节点 A 向节点 B、D 发送新数据，节点 B 收到新数据后，变成活跃节点，然后节点 B 向节点 C、D 发送新数据。其实，谣言传播非常具有传染性，它适合动态变化的分布式系统。

如何使用 Anti-entropy 实现最终一致

在分布式存储系统中，实现数据副本最终一致性，最常用的方法就是反熵了。为了帮你彻底理解和掌握在实际环境中实现反熵的方法，我想以自研 InfluxDB 的反熵实现为例，具体带你了解一下。

在自研 InfluxDB 中，一份数据副本是由多个分片组成的，也就是实现了数据分片，三节点三副本的集群，就像下图的样子：

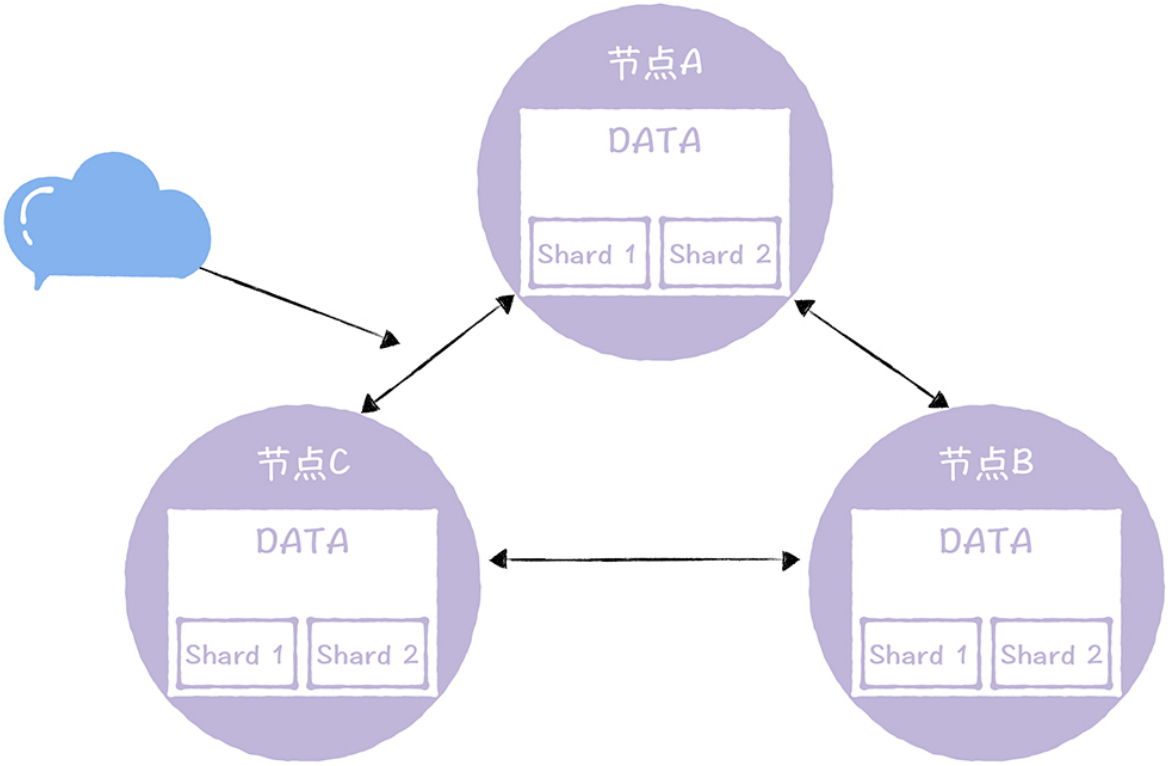


图8

反熵的目标是确保每个 DATA 节点拥有元信息指定的分片，而且不同节点上，同一分片组中的分片都没有差异。比如说，节点 A 要拥有分片 Shard1 和 Shard2，而且，节点 A 的 Shard1 和 Shard2，与节点 B、C 中的 Shard1 和 Shard2，是一样的。

那么，在 DATA 节点上，存在哪些数据缺失的情况呢？也就说，我们需要解决哪些问题呢？

我们将数据缺失，分为这样 2 种情况。

缺失分片：也就是说，在某个节点上整个分片都丢失了。

节点之间的分片不一致：也就是说，节点上分片都存在，但里面的数据不一样，有数据丢失的情况发生。

第一种情况修复起来不复杂，我们只需要将分片数据，通过 RPC 通讯，从其他节点上拷贝过来就可以了：

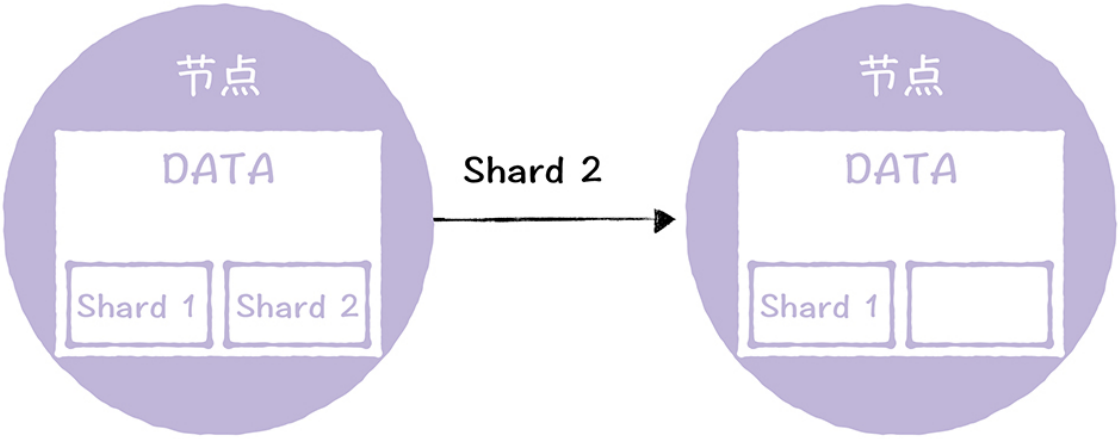


图9

你需要注意的是第二种情况，因为第二种情况修复起来要复杂一些。我们需要设计一个闭环的流程，按照一个顺序修复，执行完流程后，也就是实现了一致性了。具体是怎么设计的呢？

它是按照一定顺序来修复节点的数据差异，先随机选择一个节点，然后循环修复，每个节点生成自己节点有、下一个节点没有的差异数据，发送给下一个节点，进行修复（为了方便演示，假设 Shard1、Shard2 在各节点上是不一致的）：

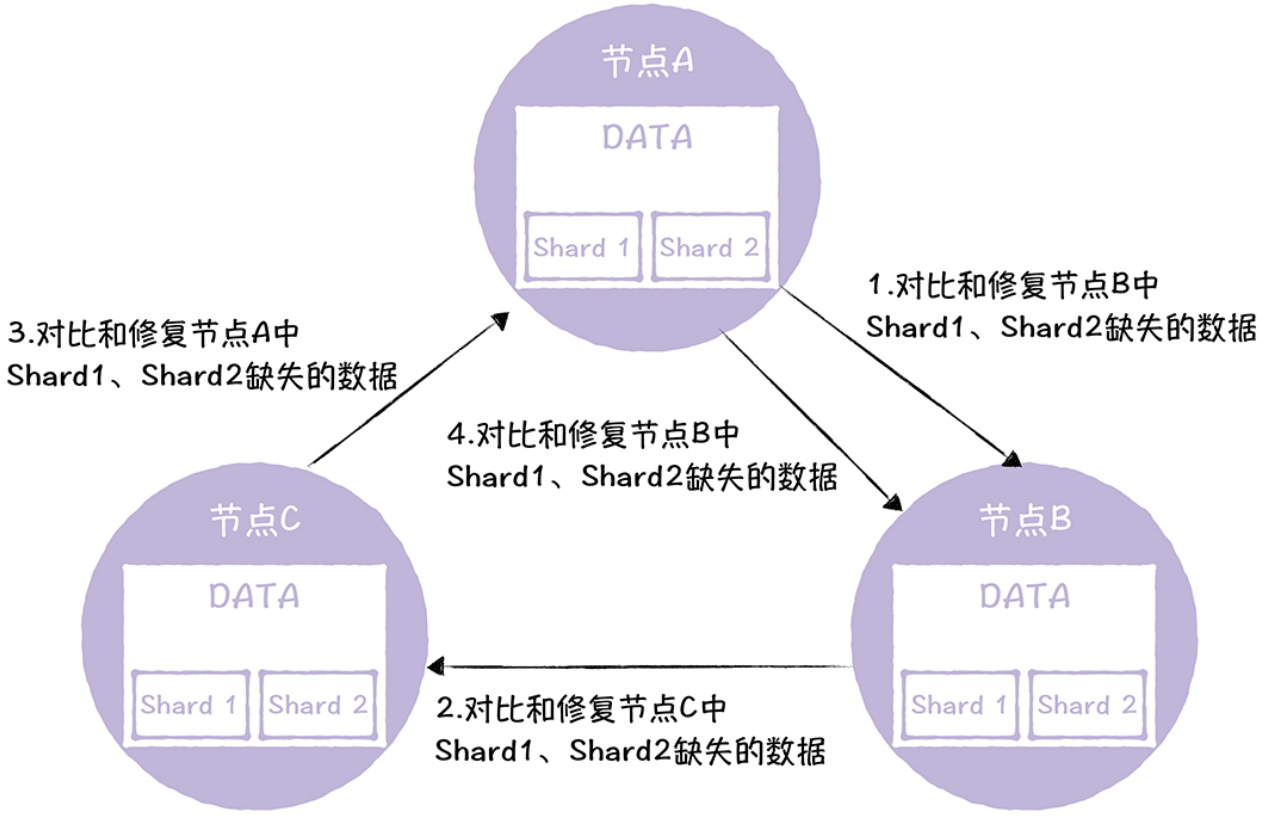


图10

从图中你可以看到，数据修复的起始节点为节点 A，数据修复是按照顺时针顺序，循环修复的。需要你注意的是，最后节点 A 又对节点 B 的数据执行了一次数据修复操作，因为只有这样，节点 C 有、节点 B 缺失的差异数据，才会同步到节点 B 上。

学到这里你可以看到，在实现反熵时，实现细节和最初算法的约定有些不同。比如，不是一个节点不断随机选择另一个节点，来修复副本上的熵，而是设计了一个闭环的流程，一次修复所有节点的副本数据不一致。

为什么这么设计呢？因为我们希望能在一个确定的时间范围内实现数据副本的最终一致性，而不是基于随机性的概率，在一个不确定的时间范围内实现数据副本的最终一致性。

这样做能减少数据不一致对监控视图影响的时长。而我希望你能注意到，技术是要活学活用的，要能根据场景特点权衡妥协，设计出最适合这个场景的系统功能。**最后需要你注意的是，因为反熵需要做一致性对比，很消耗系统性能，所以建议你将是否启用反熵功能、执行一致性检测的时间间隔等，做成可配置的，能在不同场景中按需使用。**

内容小结

以上就是本节课的全部内容了，本节课我主要带你了解了 Gossip 协议、如何在实际系统中实现反熵等。我希望你明确这样几个重点：

1. 作为一种异步修复、实现最终一致性的协议，反熵在存储组件中应用广泛，比如 Dynamo、InfluxDB、Cassandra，我希望你能彻底掌握反熵的实现方法，在后续工作中，需要实现最终一致性时，优先考虑反熵。
2. 因为谣言传播具有传染性，一个节点传给了另一个节点，另一个节点又将充当传播者，传染给其他节点，所以非常适合动态变化的分布式系统，比如 Cassandra 采用这种方式动态管理集群节点状态。

在实际场景中，实现数据副本的最终一致性时，一般而言，直接邮寄的方式是一定要实现的，因为不需要做一致性对比，只是通过发送更新数据或缓存重传，来修复数据的不一致，性能损耗低。在存储组件中，节点都是已知的，一般采用反熵修复数据副本的一致性。当集群节点是变化的，或者集群节点数比较多时，这时要采用谣言传播的方式，同步更新数据，实现最终一致。

课堂思考

既然使用反熵实现最终一致性时，需要通过一致性检测发现数据副本的差异，如果每次做一致性检测时都做数据对比的话，肯定是比较消耗性能的，那有什么办法降低一致性检测时的性能消耗呢？欢迎在留言区分享你的看法，与我一同讨论。

最后，感谢你的阅读，如果这篇文章让你有所收获，也欢迎你将它分享给更多的朋友。

© 版权归极客邦科技所有，未经许可不得传播售卖。 页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇　10 | 一致哈希算法：如何分群，突破集群的“领导者”限制?

下一篇　12 | Quorum NWR算法：想要灵活地自定义一致性，没问题!

精选留言 (17)

写留言

Jialin

2020-03-06

课堂笔记（综合第四讲和第十一讲内容）

1.Gossip 协议存在的原因?

为了实现 BASE 理论中的“最终一致性原则”。两阶段提交协议和 Raft 算法需要满足“大多数服务节点正常运行”原则，如果希望系统在少数服务节点正常运行的情况下，仍能对

外提供稳定服务，这时就需要实现最终一致性。...

展开

5

myrfy
2020-03-06

老师，gossip中，遇到数据冲突，以谁为准呢?

展开

1 2

Scott
2020-03-08

merkle tree可以用来减少比较差异负担的开销吗

展开

1

小晏子
2020-03-06

一致性检测是对比两个节点上的数据是否一致，如果每次都是全量比对，那么肯定性能不是很高，那为了提升性能，那么最好是不比对或增量数据比对，这里抛个砖，
1. 假设从某一时刻开始，所有节点数据都是一致的，每个节点都记录从这个时刻开始的数据变化，
2. 当反墒放生时，先看相关的两个节点上数据变化是否为空，如果为空就证明两个节点...

展开

作者回复: 加一颗星:)

1

忆水寒
2020-03-06

老师，有个疑惑。文章中说直接邮寄肯定要实现的，在固定节点的系统中可以实现反墒修复一致性。那么两者如何兼容呢？是不是每次有新数据写入，先执行直接邮寄。然后周期性的执行反墒修复数据一致性呢?

2 1

旅途
2020-03-10

把数据生成一个hash值 每次比较这个hash值 我之前看过微服务中拉取注册中心的配置时,

不是拉取所有的配置和本地比较 而是通过生成的hash值比较的 和这个应该是一个道理

崔伟协
2020-03-09

直接邮寄，反熵，谣言传播三者的区别是什么

展开

InfoQ_b4b7ff0241e5
2020-03-09

从网上搜了搜相关资料，发现大部分资料将谣言传播等同于gossip协议，也有把反熵等同于gossip协议的，感到很迷惑。

展开

Daiver
2020-03-07

Fabric目前1.4.1版本以上实现了raft共识，不过用的是etcd的组件；Hyperledger Fabric中组织内部的peer节点之间为了同步账本数据使用了gossip协议。

展开

作者回复: 加一颗星:)

唐明
2020-03-06

请问老师，数据可以从任意一个副本节点写入吗?

展开

作者回复: 可以的，这也是为什么需要实现anti-entropy，实现最终一致性。

朱东旭
2020-03-06

分片应该区分主分片和副本分片吧，只有主分片能更新。

vi
2020-03-06

老师，对于数据复制，只有新增或减少的两处情况，有没有相同的数据不同版本的情况出现，比如数据库中相同ID的数据，某个字段不一致，要不要利用记录的时间擢来制定一个规则，以最后更新的数据为准确数据来做更新复制

展开

Geek_8af153
2020-03-06

同问因为分区错误导致数据不一致时，合并冲突怎么解决?

panda
2020-03-06

反熵要有一个以谁为准的定义，应该有个日志编号之类的来表明吧?

约书亚
2020-03-06

内容挺重要但疑问挺多，求翻牌：
Influxdb的例子讲解了修复数据，但没讲数据一开始是怎样产生的。结合最后总结中提到“直接邮寄必须实现”，能不能这样猜测：
一个请求到达某实例，实例写入本机shard，然后通过类似“直接邮寄”，发送给其他shard副本。修复工作是在在后台异步进行的。...

展开

myrfy
2020-03-06

参考区块链思想，系统内维护一个当前版本号，也就是区块链中的当前高度。因为是内部系统，可以暂时不考虑安全问题，认为大家都是诚实的，这样就可以去掉区块链中的POW任意节点只要发现自己当前高度和网络中接受到的高度不一致，就更新自己的状态

展开

每天晒白牙
2020-03-06

今日得到
出于可用性考虑(即使一个节点也照常运行)，而二阶段提交和 raft 协议都是需要保证大多数几点正常运行才能保证稳定运行。根据 base 理论要实现最终一致性，所以 gossip 协议就出来了。
…
展开